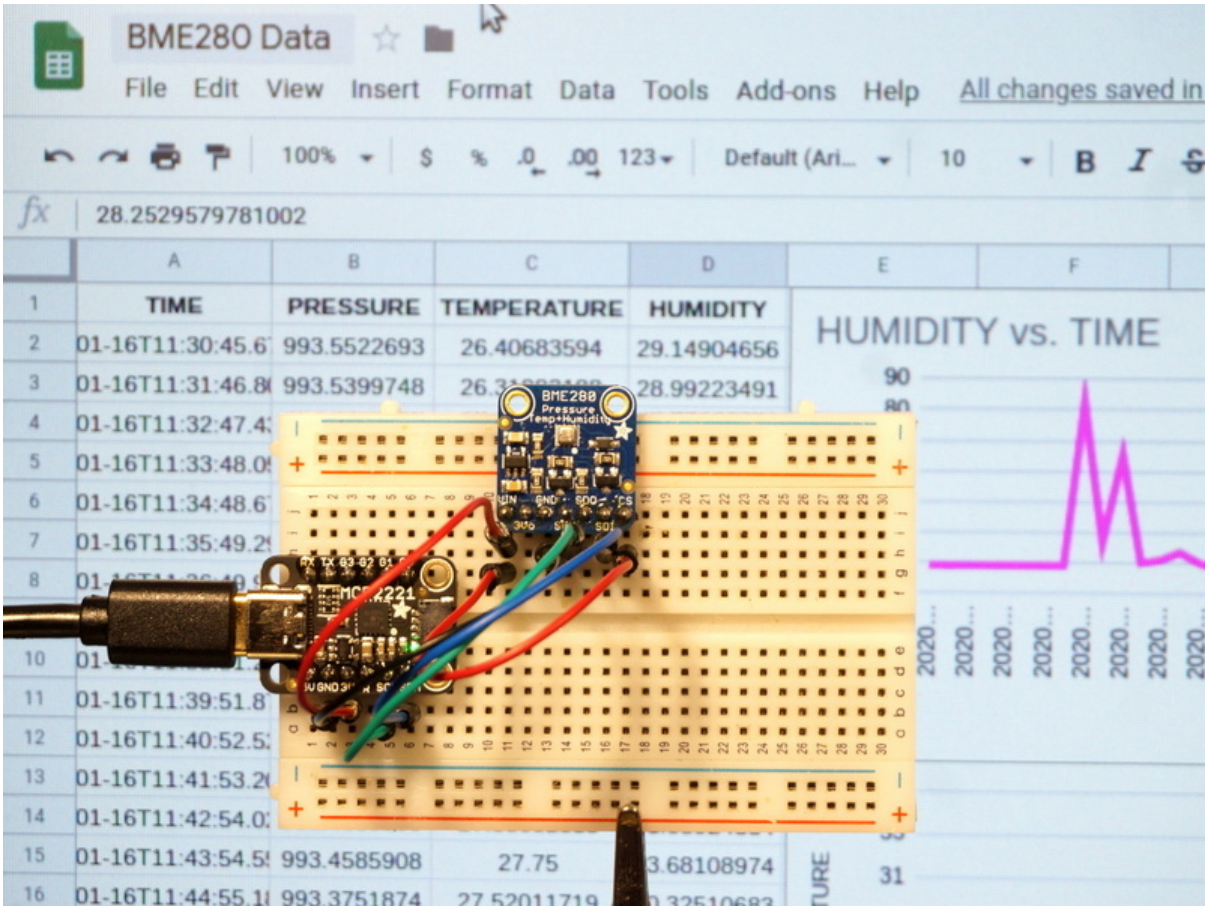
Google Docs Sensor Logging From Your PC

Created by Carter Nelson

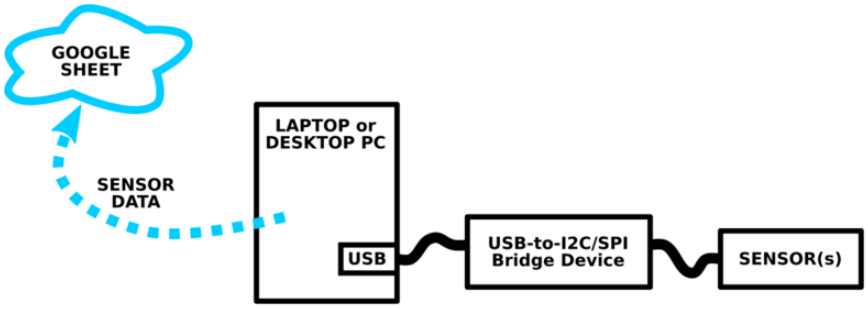


https://learn.adafruit.com/gdocs-sensor-logging-from-your-pc

Last updated on 2024-06-03 03:00:06 PM EDT

Table of Contents

Overview

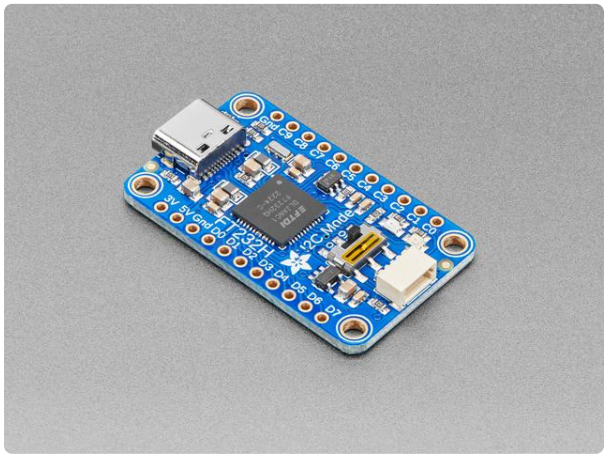


This guide will show you how you may log data from an I2C or SPI based sensor to a Google Sheet, directly from your PC, using Python.

No microcontrollers are required - we read data directly into a computer!

To enable reading I2C or SPI based sensors from your PC, we'll use a special USB bridge device. There are several options:

Adafruit MCP2221A Breakout - General Purpose USB to GPIO ADC I2C
Wouldn't it be cool to drive a tiny OLED display, read a
https://www.adafruit.com/product/4471



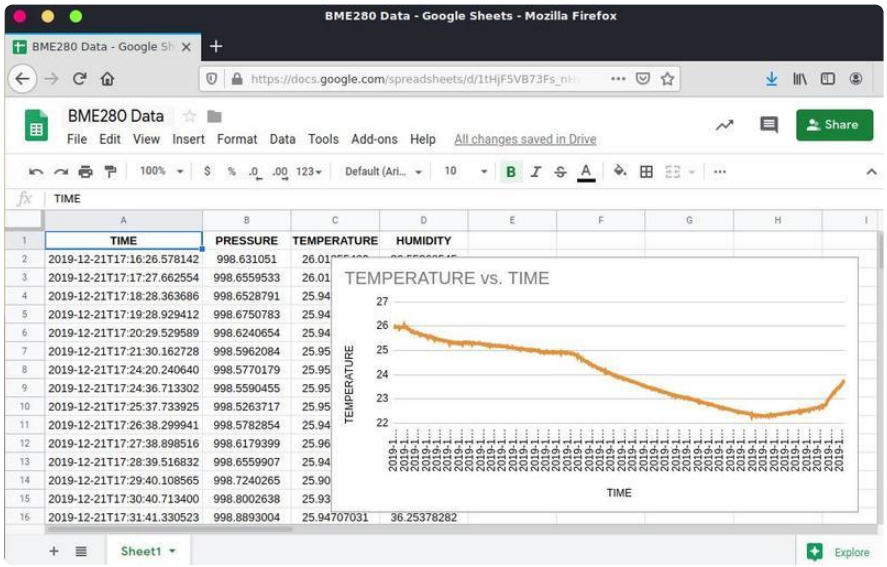
Adafruit FT232H Breakout - General Purpose USB to GPIO, SPI, I2C
Wouldn't it be cool to drive a tiny OLED display, read a
https://www.adafruit.com/product/2264

[Binho Nova Multi-Protocol USB Host Adapter]
Discontinued - you can grab our Adafruit Trinkey QT2040 - RP2040 USB...
https://www.adafruit.com/product/4459

And then we can install Blinka (https://adafru.it/BSN) to allow us to use CircuitPython libraries (https://adafru.it/ENC) to make using the sensor easy.

There's a fair amount of initial software setup that must be done, most of which is for setting up services and credentials on Google. But once everything is in place, sending sensor data to a Google Sheet is really easy.

Let's get started...



Software Setup

There are three main pieces to get this all working:

- **Google Setup -** do this in your Google account to enable features and create credentials you can use to access your Google Docs.
- **Python Setup -** install libraries to allow accessing your docs using credentials created above

- **Sensor Setup** - install necessary software to access I2C/SPI sensors via USB

We'll go through each of these separately in the following three sections.

Google Setup

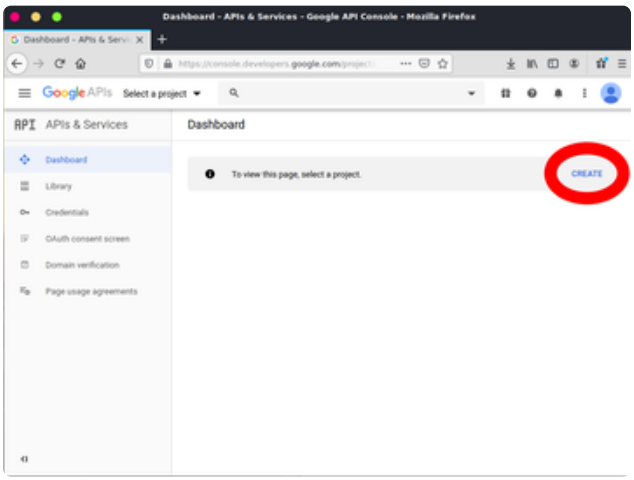
These steps are taken via a web browser while logged in to your Google user account.

This is probably the trickiest part. It's really just a matter of going to your Google account to enable features and create credentials, which you will then use in your Python program to access your Google Docs. But the specific details, like the user interface, menu locations, specific names for things, etc. seem to always be changing. So getting through this part can be a bit of a slog.

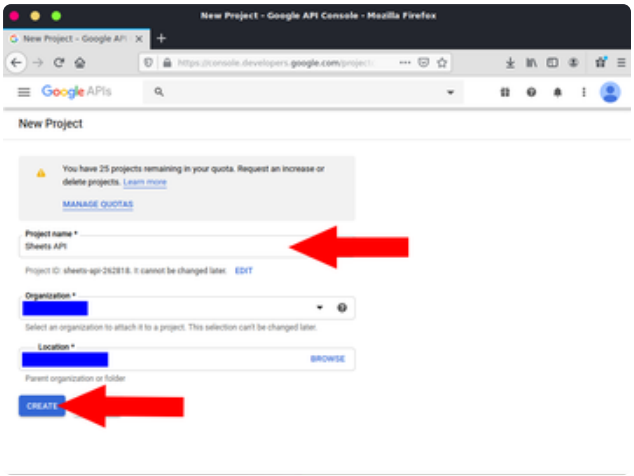
But here we go...

Create a New Project

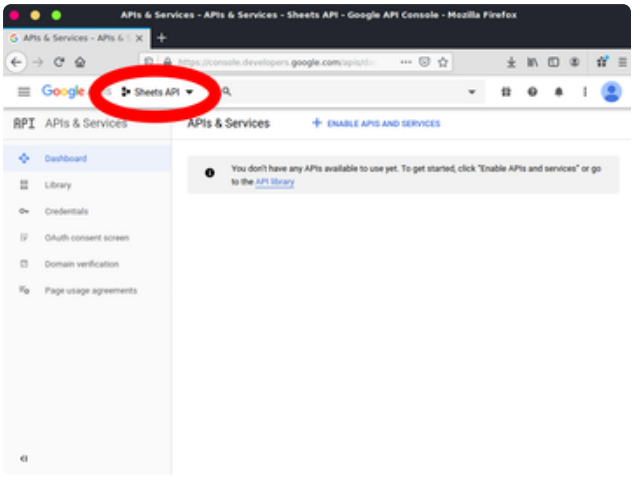
Follow these steps to create a new project.



From your Google API Dashboard (https://adafru.it/lfQ), click **CREATE**
NOTE: If you have already existing projects, this screen will look different.



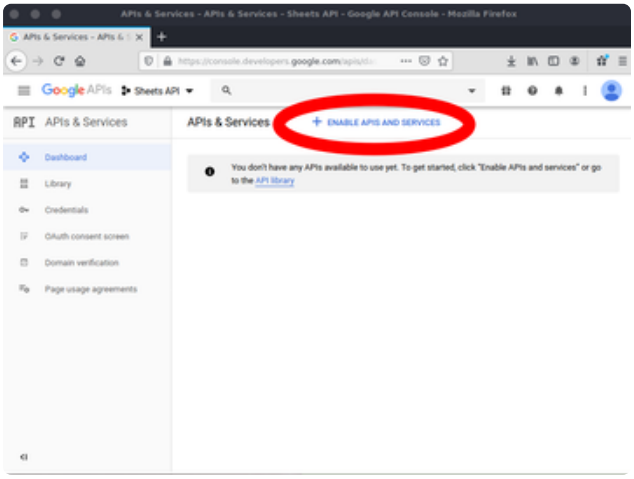
Give the project a name, like **Sheets API**
You can leave **Organization** and **Location** blank or with default values
Click **CREATE**



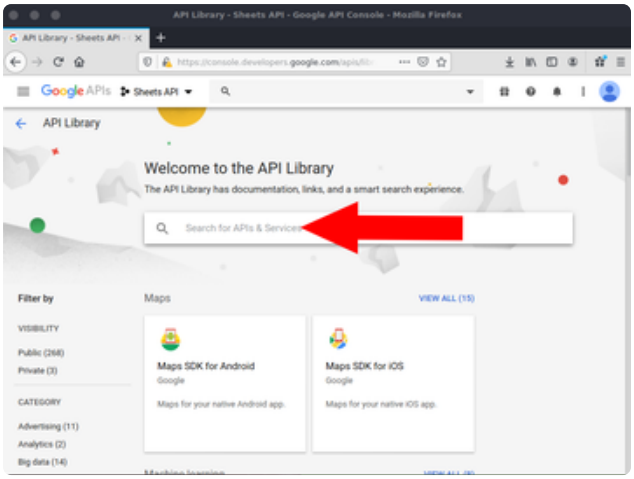
You should be returned to the main Dashboard screen and your new project is selected.

Enable APIs for the Project

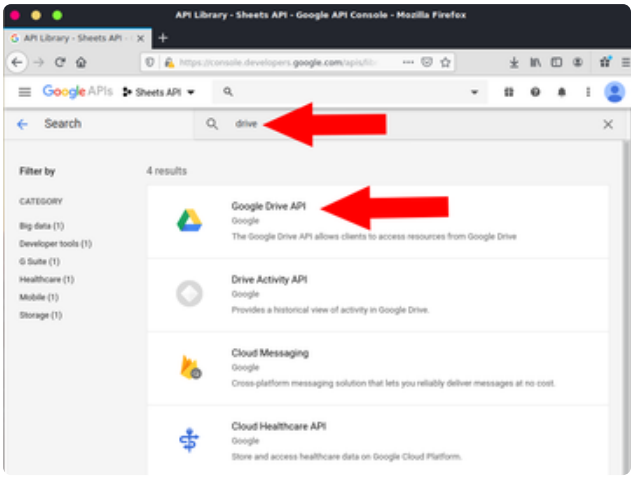
Follow these steps to add the needed APIs to the Project.



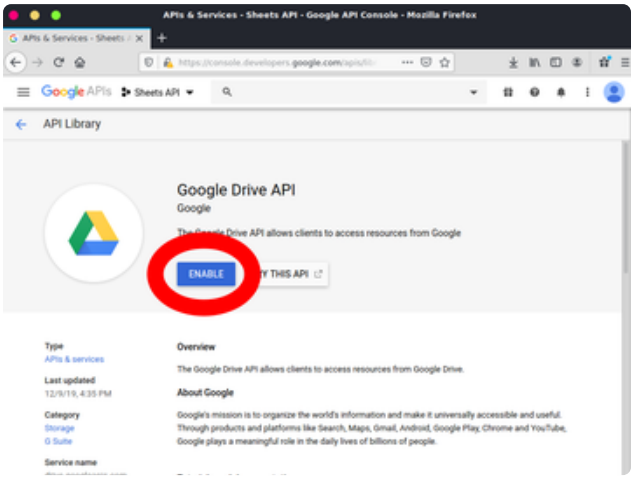
From the Dashboard screen, click **ENABLE APIS AND SERVICES**



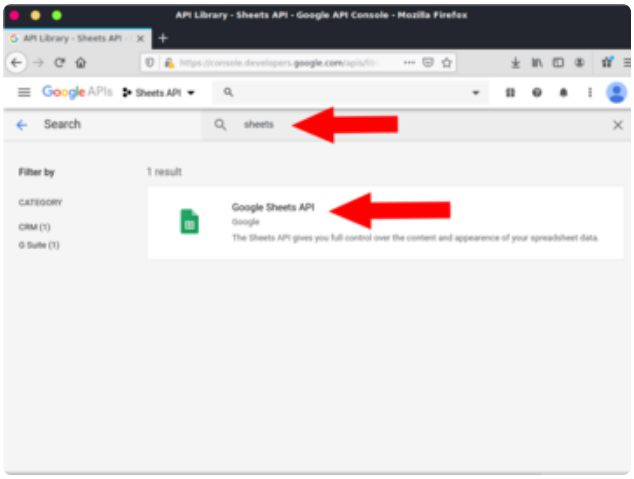
In the Search box, enter "drive"



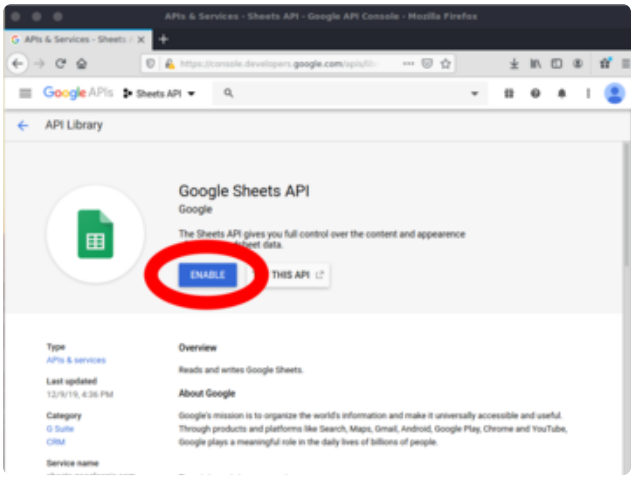
After entering "drive" in the Search box, select **Google Drive API** from the list.



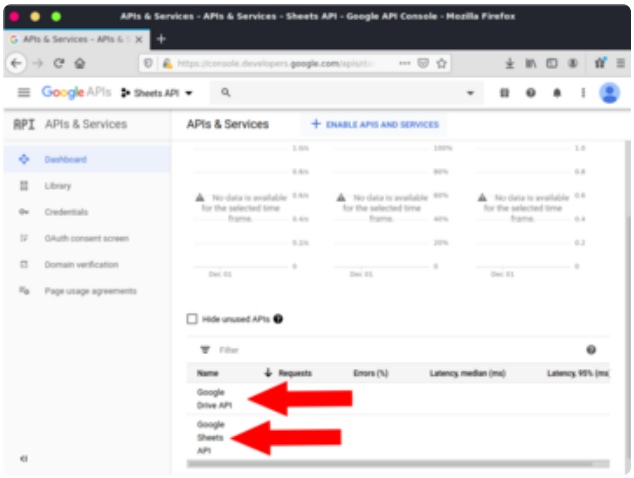
On the next screen, click **ENABLE**.



Now search for "sheets" and select **Google Sheets API**



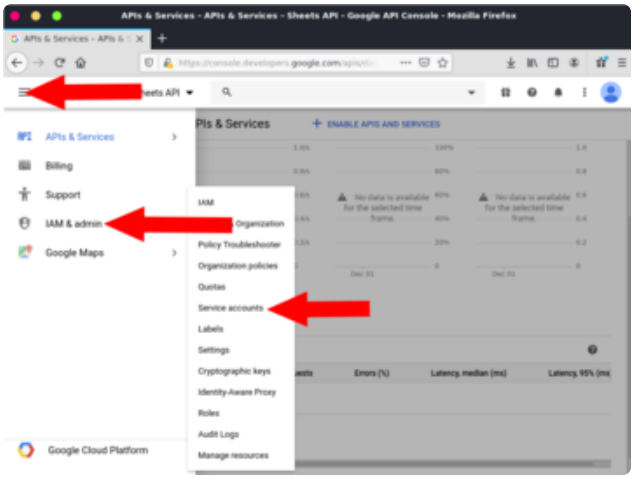
On the next screen, click **ENABLE**.



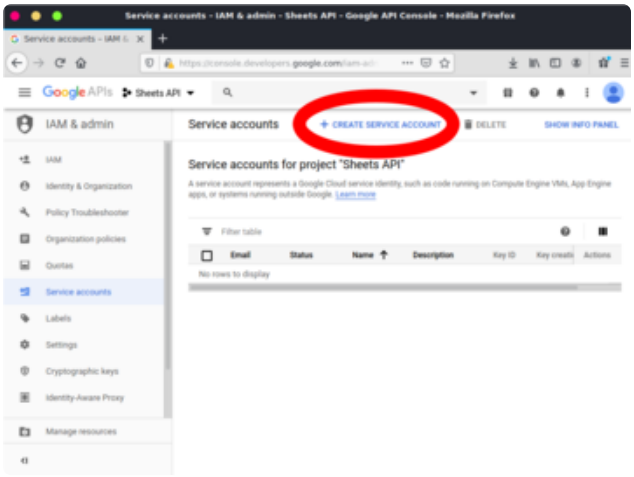
Back at the main Dashboard, verify that the two APIs have been added.

Create a Service Account

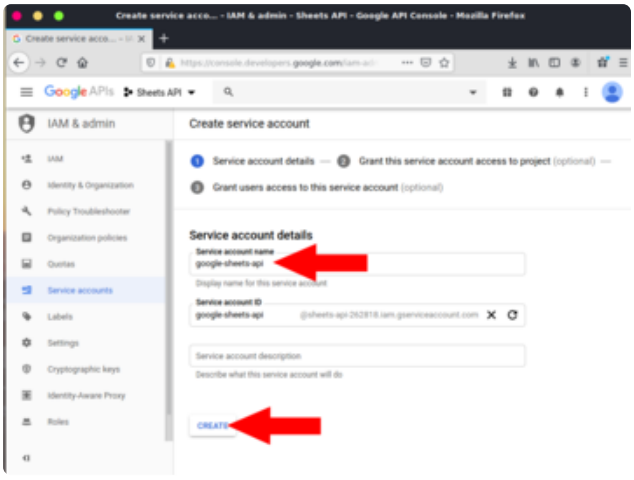
Follow these steps to create the credentials needed to access your Sheet from Python.



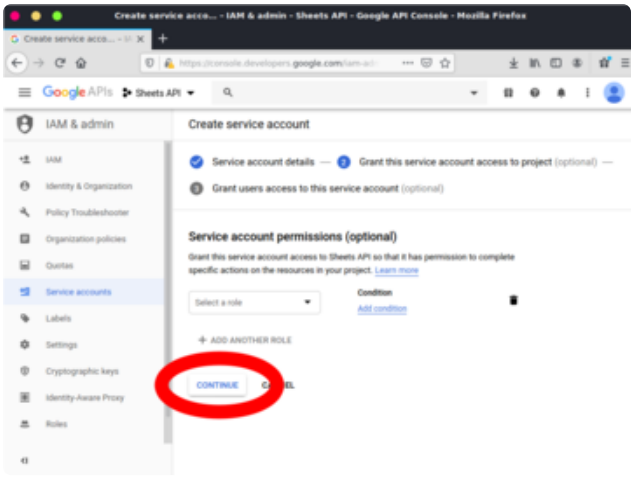
Click on the 3 bars to access the menu and go to **IAM & admin -> Service accounts**



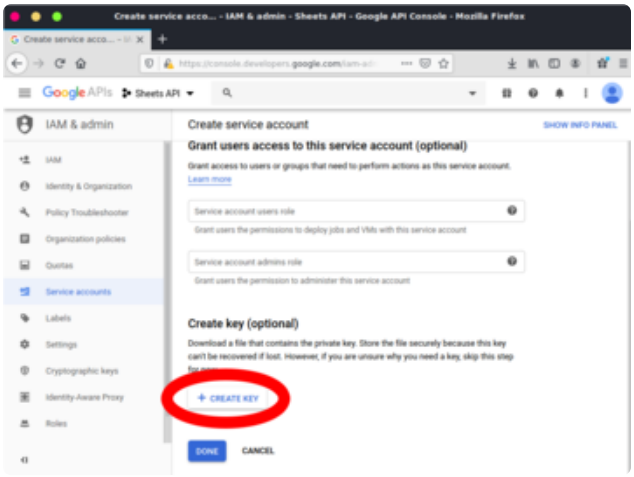
On the Service Accounts screen, click **CREATE SERVICE ACCOUNT**.



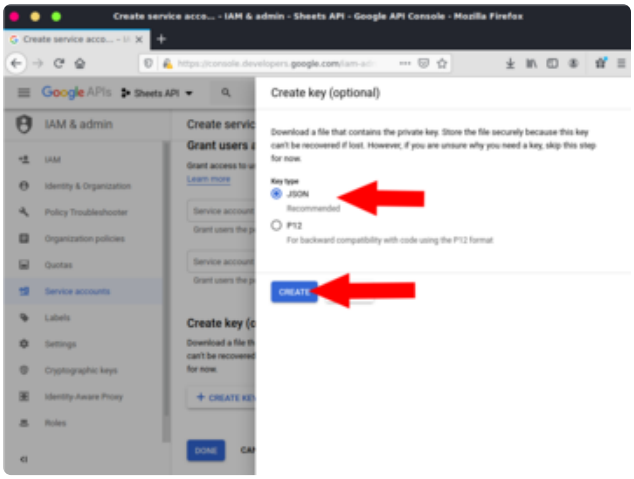
Give the service account a name like **google-sheets-api** and click **CREATE**.



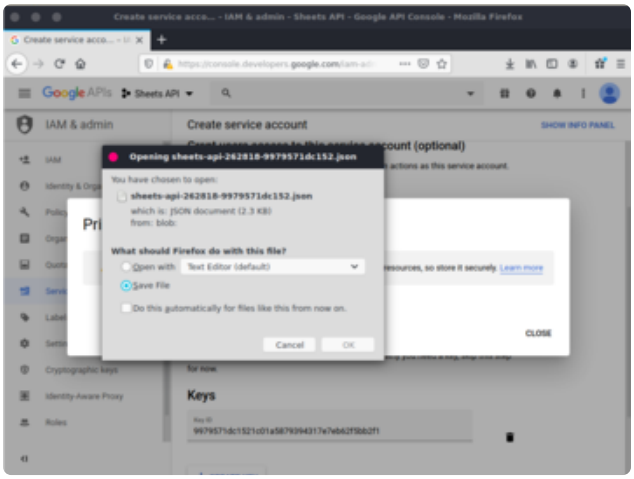
Click **CONTINUE** on the next screen.



On the final screen, click **CREATE KEY**.



Make sure **JSON** is selected and click **CREATE**.



You will be prompted to save a **.json** file. Save this somewhere safe.

The .json file you download contains secret information. Be sure to keep it safe and don't share the contents with anyone. Be careful to not post the credentials on GitHub or similar sites.

Everything should now be enabled and set up.

If you have created and saved the .json file successfully, then move on to the next step.

Python Setup

Here we install the required Python libraries and create a test Sheet to make sure everything works. More information can be found in the Google documentation here:

Google Docs API Python Quickstart

https://adafru.it/lfR

Download Python (mainly for Windows Users)

Python 3 comes standard on Linux and macOS. But Windows users likely do not have it installed as a default. If you need Python and the pip installer, go to https://www.python.org/downloads/ (https://adafru.it/fa7) and download the latest version of Python 3.x for your operating system.

Install Python Libraries

We need two libraries:

- google-auth-oauthlib (https://adafru.it/lfS) - needed for authentication
- google-api-python-client (https://adafru.it/lfT) - the main Google API client library

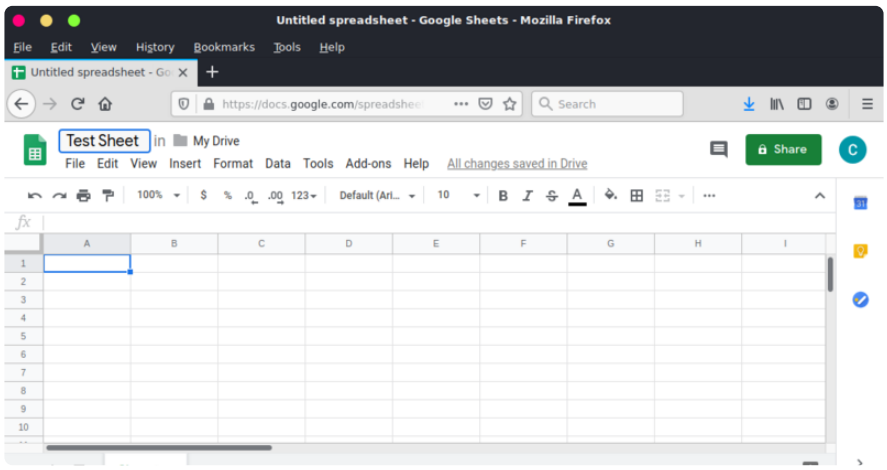
which we can install with pip (which is installed with Python 3). Go to a command prompt and type:

```
pip3 install google-api-python-client google-auth-oauthlib
```

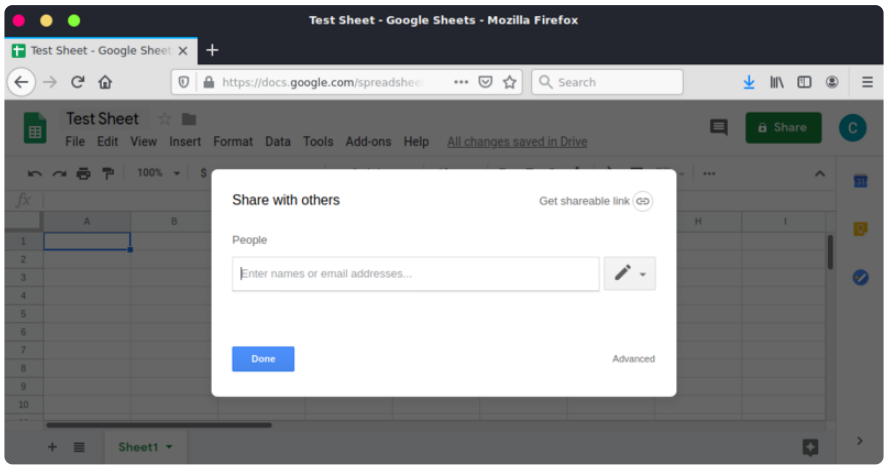
Create A Test Sheet

To test everything, let's go ahead and create a Sheet.

Log in to your Google account and go to Docs and create a new Sheet and give it a name like **Test Sheet**:



Under **File -> Share**, add the `"client_email"` address from the **.json** file created in the previous step. You can open the **.json** file in a text editor to find this value.



You'll also need the `spreadsheetID` which you can get from the URL. It is the big long gibberish value between "/d/" and "/edit". (more info (https://adafru.it/lfU))

Run Example

Then, try running this example:

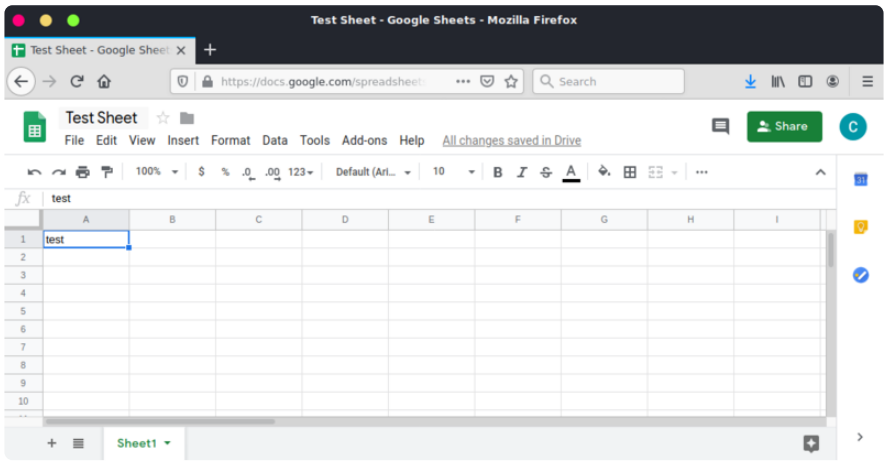
You need to edit this example with your specific info. See below.

```
from google.oauth2.service_account import Credentials
from googleapiclient.discovery import build

SERVICE_ACCOUNT_FILE = 'YOUR_CREDENTIALS_FILE.json'
SPREADSHEET_ID = 'YOUR_SPREADSHEET_ID'
SCOPES = ['https://spreadsheets.google.com/feeds',
          'https://www.googleapis.com/auth/drive']

credentials = Credentials.from_service_account_file(SERVICE_ACCOUNT_FILE,
scopes=SCOPES)

service = build('sheets', 'v4', credentials=credentials)

sheet = service.spreadsheets()

range = 'A1'
values = [ ['test'], ]
body = {'values' : values}

sheet.values().append(spreadsheetId=SPREADSHEET_ID,
                      valueInputOption='RAW',
                      range=range,
                      body=body).execute()
```

Replace `YOUR_CREDENTIALS_FILE` with the actual name of your **.json** file and `YOUR_SPREADSHEET_ID` with what you got from the URL. Save the above as something like **gspread_test.py** and then try running it:

```
python3 gspread_test.py
```

It should run without any errors. If you then go look at your spreadsheet, you should see **test** in cell **A1**, like this:



If any errors do occur, try and troubleshoot them. Hopefully they will contain helpful messages.

If the above test worked, move on to the next step.

Sensor Setup

To get sensor data into your PC, we will use a special USB gadget to provide the necessary hardware interface. There are several options available. Go to the appropriate guide below for the device you are using.

MCP2221A (I2C)

https://adafru.it/IfV

FT232H (SPI or I2C)

https://adafru.it/FWD

Binho Nova (SPI or I2C)

https://adafru.it/IfW

You will end up installing Blinka (https://adafru.it/EQa), which the guides cover. Each of those guides also has a simple test to make sure you can talk to a sensor. You can also look at the guide specific to your sensor for additional help with wiring and additional code examples.

Once you are talking to your sensor, move on to the next step.

Sensor Logging

OK, let's put all the pieces together and log some sensor data. At this point you should have:

- Created a Google project and enabled Drive and Sheets API
- Created a .json credentials file and saved it locally
- Created a Google Sheet and shared it with the email address from the .json file
- Installed the Python libraries `google-api-python-client` and `google-auth-oauthlib`
- Installed Blinka to allow using CircuitPython libraries
- Installed the CircuitPython library for your specific sensor

The following examples show usage for two different USB bridge options as well as for I2C and SPI.

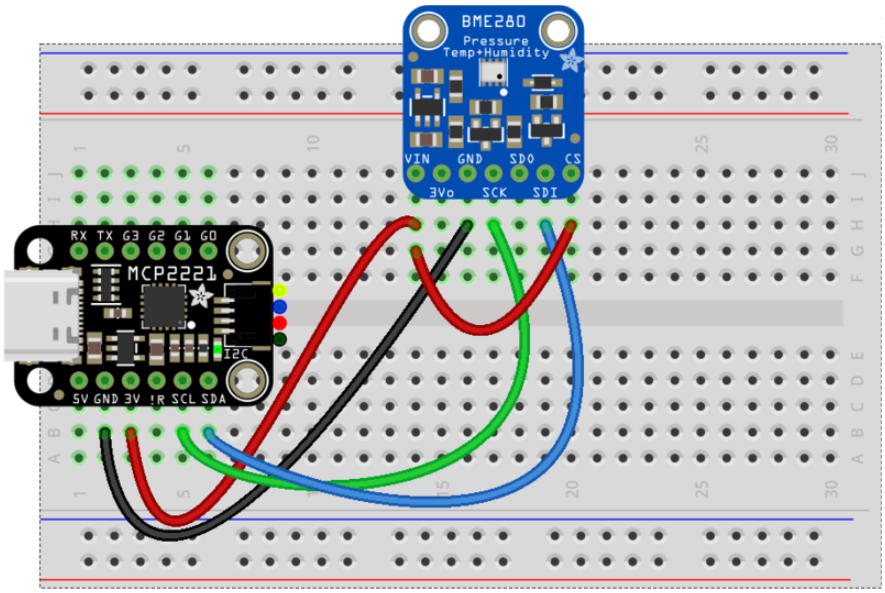
MCP2221 with BME280 on I2C

This example uses a MCP2221 with a BME280 pressure/temperature/humidity sensor accessed over an I2C wired connection. Links to setup the MCP2221 were provided early in the guide. To install the library for the BME280, see here:

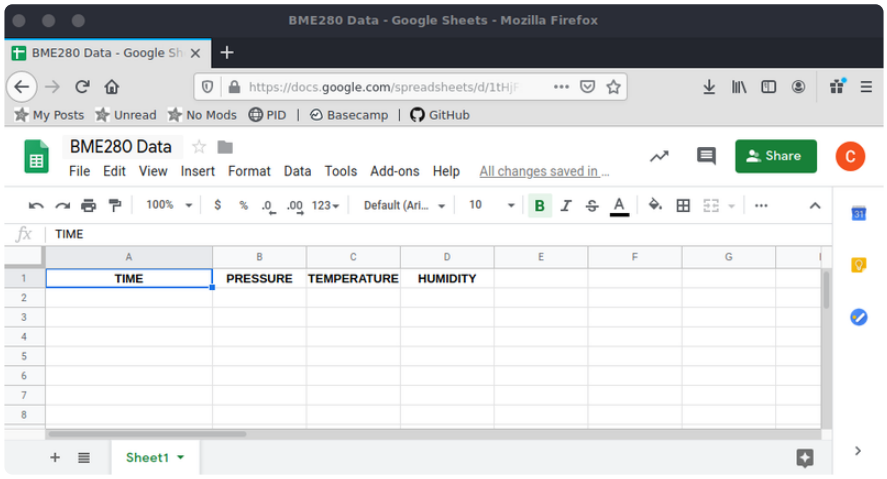
Python Installation of BME280 Library

https://adafru.it/FUP

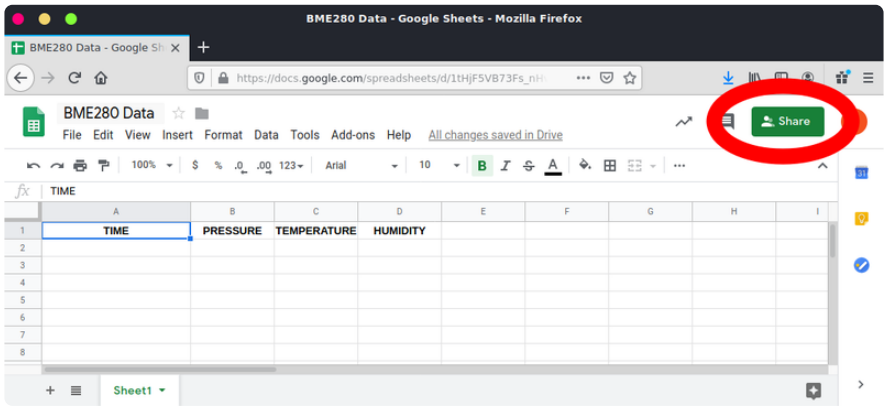
Here is the wiring diagram:



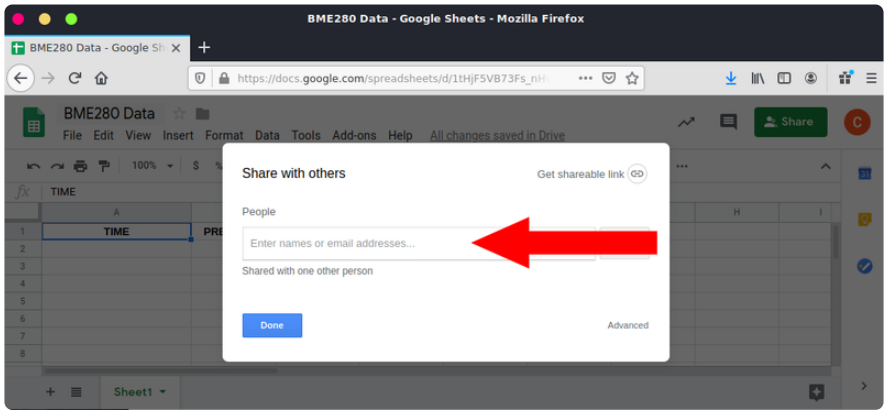
Create a new Sheet to hold this data and give it a name. Here we also add some column titles:



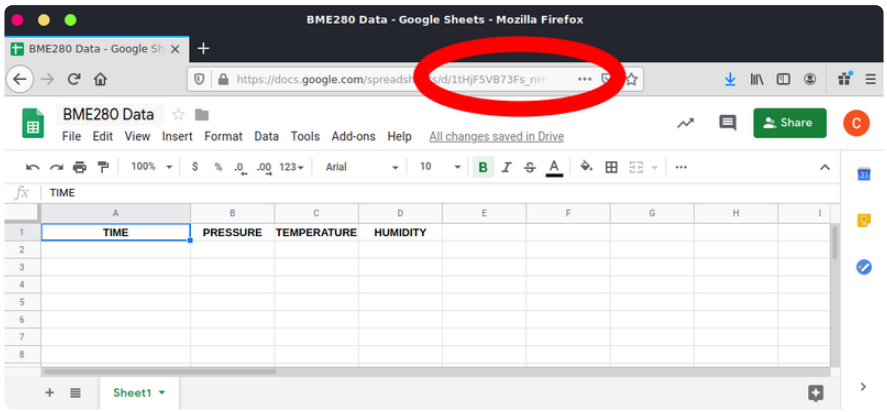
Add just like we did in the test example, share the sheet with the `"client email"` address from the **.json** file. Click the **SHARE** button:



And add the `"client email"` address from the **.json** file.



Also, grab the gibberish `spreadsheetID` from the URL so you can use it in the code below. See previous section of guide for more info on this.



And here's the code for logging BME280 sensor data to you Google Sheet. **Be sure to replace `YOUR_CREDENTIALS_FILE` and `YOUR_SHEET_ID` with the actual values for your setup.**

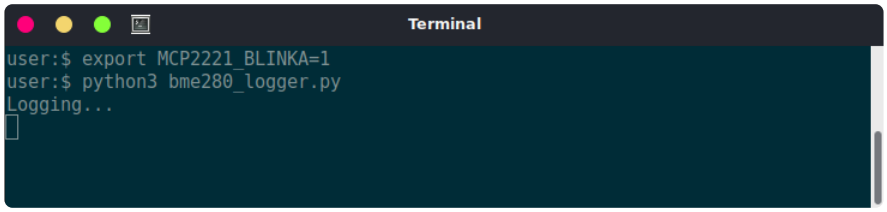
```
# SPDX-FileCopyrightText: 2020 Carter Nelson for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
from datetime import datetime
import board
import adafruit_bme280
from google.oauth2.service_account import Credentials
from googleapiclient.discovery import build

#--| User Config |-------------------------------------------------
SERVICE_ACCOUNT_FILE = 'YOUR_CREDENTIALS_FILE.json'
SPREADSHEET_ID = 'YOUR_SHEET_ID'
DATA_LOCATION = 'A1'
UPDATE_RATE = 60
#--| User Config |-------------------------------------------------

# Sensor setup
i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
bme = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# Google Sheets API setup
SCOPES = ['https://spreadsheets.google.com/feeds',
          'https://www.googleapis.com/auth/drive']
CREDS = Credentials.from_service_account_file(SERVICE_ACCOUNT_FILE, scopes=SCOPES)
SHEET = build('sheets', 'v4', credentials=CREDS).spreadsheets()

# Logging loop
print("Logging...")
while True:
    values = [[datetime.now().isoformat(), bme.pressure, bme.temperature,
bme.humidity]]
    SHEET.values().append(spreadsheetId=SPREADSHEET_ID,
                          valueInputOption='RAW',
                          range=DATA_LOCATION,
                          body={'values' : values}).execute()
    time.sleep(UPDATE_RATE)
```

Don't forget to also set the BLINKA_MCP2221 environment variable.

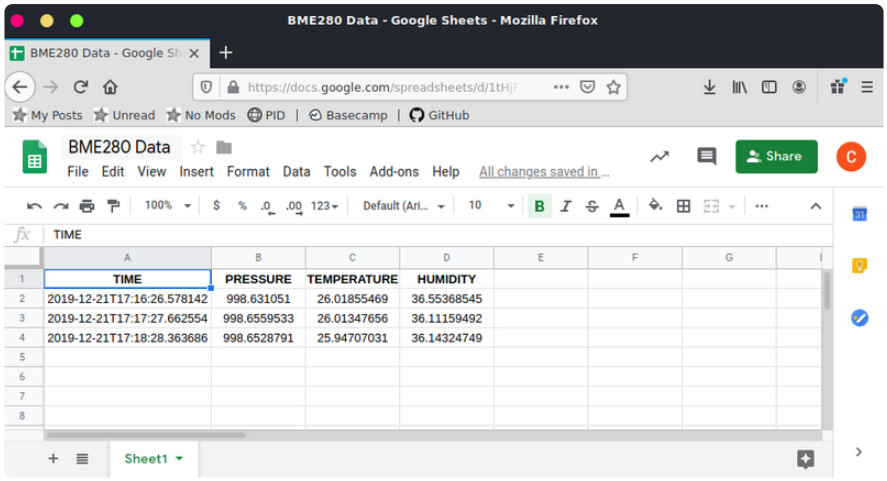
Save that as something like **bme280_logger.py** and then run it with:

```
python3 bme280_logger.py
```

It should run without errors and start logging:



After a couple of minutes, go back and look at your sheet. It should be updated with new values:



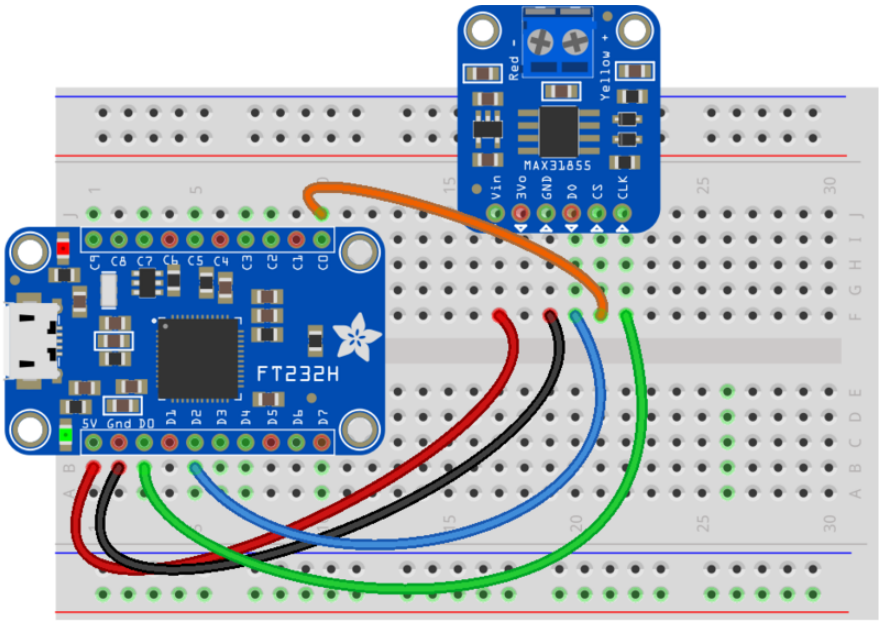
FT232H with MAX31855 on SPI

This example uses a FT232H with a MAX31855 thermocouple breakout accessed over SPI. Links to setup the FT232H were provided early in the guide. To install the library for the MAX31855, see here:

Python Installation of MAX31855 Library

https://adafru.it/IfX

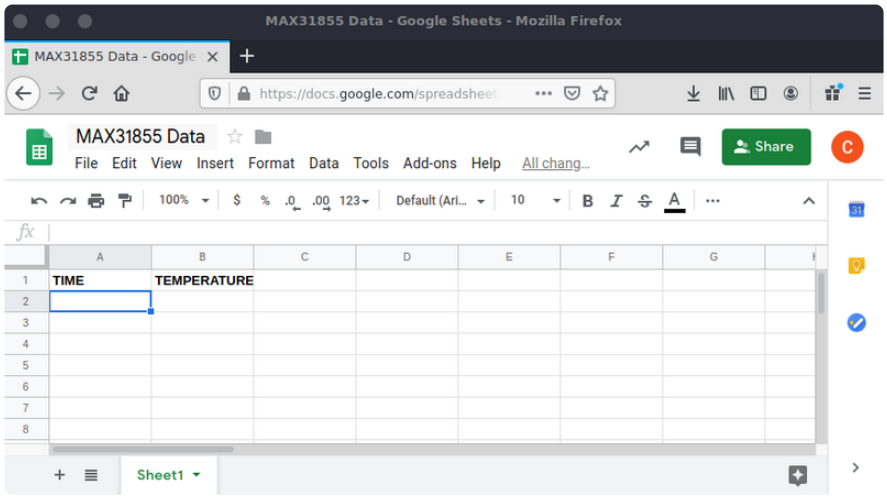
Here is the wiring diagram:



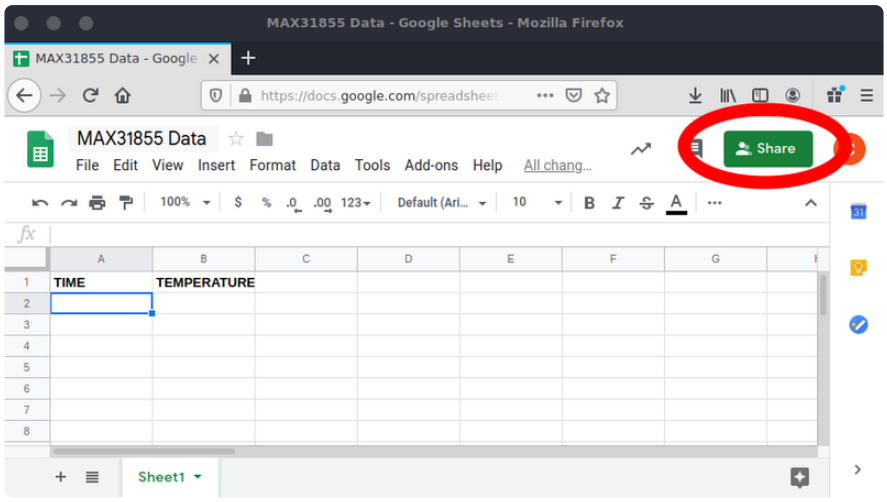
Sensor

You will also need an appropriate thermocouple sensor attached to the MAX31855 **Red -** and **Yellow +** terminals. See the Adafruit thermocouple guide (https://adafru.it/lkA) in selecting the appropriate sensor.

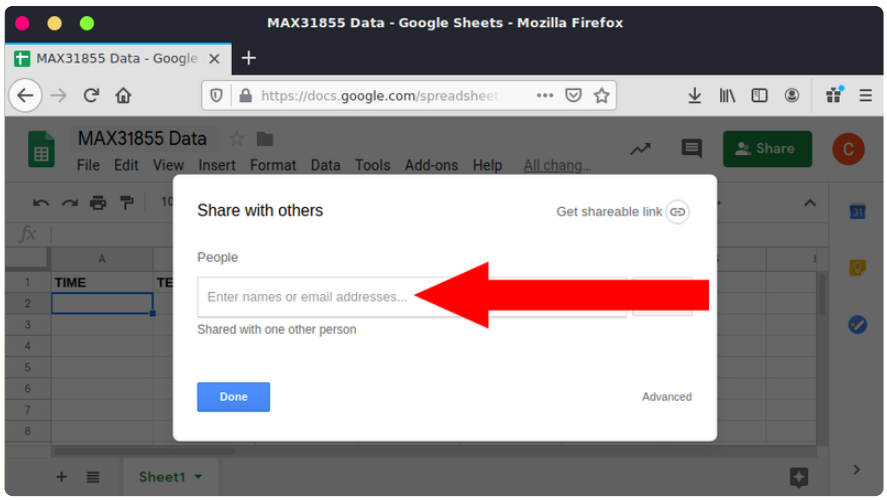
Create a new Sheet to hold this data and give it a name. Here we also add some column titles:



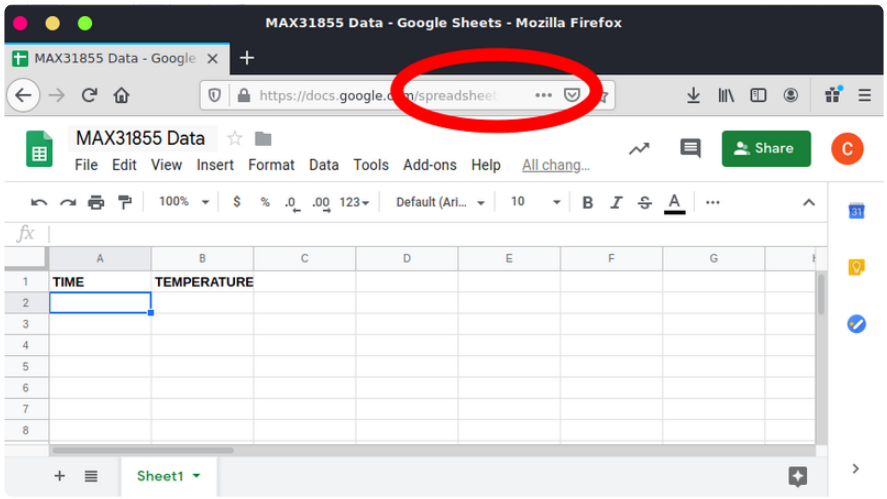
Add just like we did in the test example, share the sheet with the `"client email"` address from the **.json** file. Click the **SHARE** button:



And add the `"client email"` address from the **.json** file.



Also, grab the gibberish `spreadsheetID` from the URL so you can use it in the code below. See previous section of guide for more info on this.



And here's the code for logging MAX31855 sensor data to you Google Sheet. **Be sure to replace `YOUR_CREDENTIALS_FILE` and `YOUR_SHEET_ID` with the actual values for your setup.**

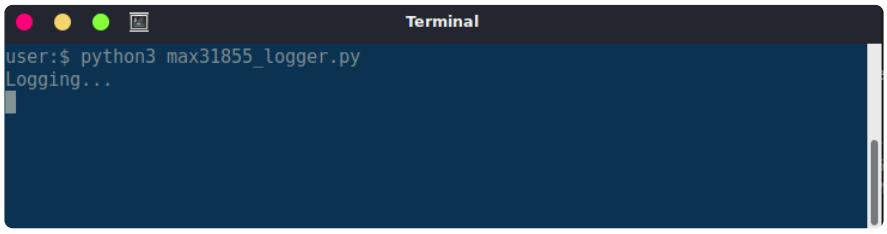
```
# SPDX-FileCopyrightText: 2020 Carter Nelson for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
from datetime import datetime
import board
import digitalio
import adafruit_max31855
from google.oauth2.service_account import Credentials
from googleapiclient.discovery import build

#--| User Config |-------------------------------------------------
SERVICE_ACCOUNT_FILE = 'YOUR_CREDENTIALS_FILE.json'
SPREADSHEET_ID = 'YOUR_SHEET_ID'
DATA_LOCATION = 'A1'
UPDATE_RATE = 60
#--| User Config |-------------------------------------------------

# Sensor setup
cs = digitalio.DigitalInOut(board.C0)
max31855 = adafruit_max31855.MAX31855(board.SPI(), cs)

# Google Sheets API setup
SCOPES = ['https://spreadsheets.google.com/feeds',
          'https://www.googleapis.com/auth/drive']
CREDS = Credentials.from_service_account_file(SERVICE_ACCOUNT_FILE, scopes=SCOPES)
SHEET = build('sheets', 'v4', credentials=CREDS).spreadsheets()

# Logging loop
print("Logging...")
while True:
    values = [[datetime.now().isoformat(), max31855.temperature]]
    SHEET.values().append(spreadsheetId=SPREADSHEET_ID,
                          valueInputOption='RAW',
                          range=DATA_LOCATION,
                          body={'values' : values}).execute()
    time.sleep(UPDATE_RATE)
```

Don't forget to also set the BLINKA_FT232H environment variable.

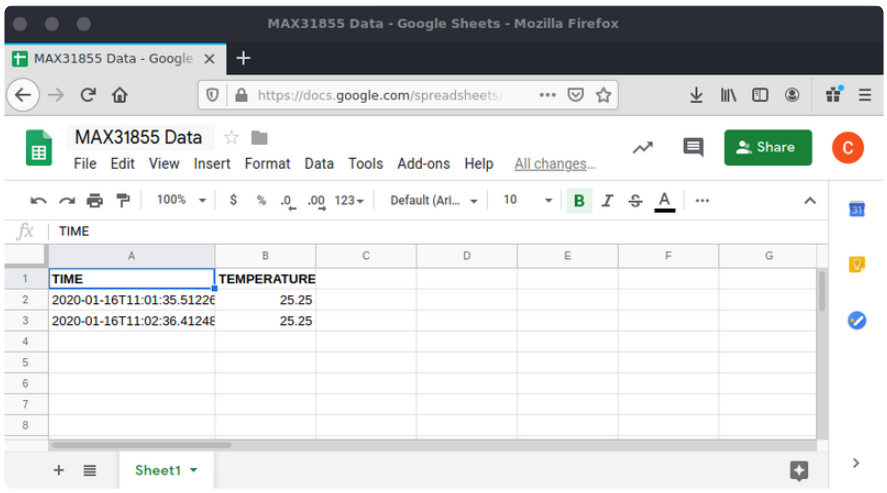
Save that as something like **max31855_logger.py** and then run it with:

```
python3 max31855_logger.py
```

It should run without errors and start logging:



After a couple of minutes, go back and look at your sheet. It should be updated with new values:


MAX31855 Data - Google Sheets - Mozilla Firefox
MAX31855 Data - Google
https://docs.google.com/spreadsheets/
MAX31855 Data
File Edit View Insert Format Data Tools Add-ons Help All changes...
Share
TIME
TIME
TEMPERATURE
2020-01-16T11:01:35.51226
25.25
2020-01-16T11:02:36.41248
25.25
Sheet1